4

Improvements in supervised BRAINNE: A method for symbolic datamining using neural networks

T.S. Dillon, T. Hossain, W. Bloomer, M. Witten
Expert and Intelligent Systems Laboratory
Applied Computing Research Institute
& School of Computer Science and Computer Engineering
La Trobe University, Bundoora Victoria 3083
Australia
Tel: +61 3 9479 2598 Fax: +61 3 9479 3060
[tharam, hossain, bloomer, mary]@cs.latrobe.edu.au

Abstract

The BRAINNE method is a technique for extracting symbolic production rules, concepts and concept hierarchies from neural networks. Previous work reported on the extraction of conjunctive and disjunctive rules for the case of binary, discrete and continuous features. In this paper we explain three new improvements. The first improvement uses a hybrid two layer network for learning disjunctive rules, where the first layer consists of a network that uses unsupervised learning and the second layer uses supervised learning. The second improvement examines the effect on the generalisation capability of the rules developed by avoiding overtraining of the neural network, and the third development is an improved approach to dealing with continuous features which greatly increases the generalisation capability of the derived rules.

Keywords

Neural networks, hybrid systems, symbolic knowledge extraction

Data Mining and Reverse Engineering S. Spaccapietra & F. Maryanski (Eds.)

1 INTRODUCTION

Neural networks can be distinguished by the type of learning employed. Supervised learning involves data sets with input patterns and their related targets or outputs. The corresponding target output values are available to guide the learning. A single layer perceptron trained with the delta rule is a supervised network. Unsupervised learning aims to detect underlying statistical patterns in the input data. For example, the self organising Kohonen network groups into clusters the output nodes that are topologically close to each other and that respond in a similar manner to the training set.

One criticism of neural networks for datamining is that the knowledge embedded in these subsymbolic systems is not easy for humans to comprehend. It is a worthwhile aim to transform the cognitive black box neural network into a cognitive clear box, if required. In other words, if the subsymbolic knowledge representation of neural networks can be mapped into a useful, understandable symbolic form then one of the constant criticisms of neural networks will no longer be relevant.

In the literature, one finds several methods for extracting propositional conjunctive rules (Fu, 1991; Gallant, 1993; Saito & Nakano, 1988; Towell & Shavlik, 1993). These essentially view rule extraction as a form of breadth first search. One difficulty here is that the computational complexity is exponential in the features. In addition, they primarily identify a conjunctive rule for each output. An important departure from this is the approach of Craven and Shavlik (1993). Their M of N method derives rules in the form:

IF (M of the following N antecedents are true) THEN ... .

If M = N then only conjunctive rules are found, and if M < N then a number of disjunctive rules are formed. They limit their consideration to networks that have discrete output classes and input features that have Boolean or nominal values.

Sestito and Dillon (1989, 1990a-c, 1991a-b, 1992, 1993, 1994) detail the research that has lead to the BRAINNE (Building Representations for AI using Neural NEtworks) method. For supervised networks, BRAINNE can successfully extract production rules from domains with both symbolic discrete attributes and continuous numeric attributes. A method of extracting rules, concepts and hierarchies has also been developed for data where no target outputs are given using unsupervised BRAINNE (Dillon et al., 1993, 1994; Sestito & Dillon, 1994).

There are two types of production rules which may be extracted; conjunctive and disjunctive. When the antecedent X consists of premises x_1, x_2, ..., x_n joined by the $\wedge$ (AND) connective, the rule is conjunctive:

$$x_1 \wedge x_2 \wedge \ldots \wedge x_n \Rightarrow C.$$

Such rules require all premises to hold for the conclusion to be true.

Disjunctive rules arise when more than one set of conditions lead to the same conclusion. The disjunctive rule

$x_1 \vee x_2 \vee \ldots \vee x_n \Rightarrow C$

comprising groups of premises joined by the $\vee$ (OR) connective, is satisfied when at least one premise is true. Note that the premises in a disjunctive rule may themselves contain several conjunctive rules.

Domains requiring disjunctive rules have two or more different sets of defining attributes for the one concept. The underlying strategy in Supervised BRAINNE is to divide the examples in the training set into subsets or partitions that can be uniquely described by one conjunctive rule. This approach requires retraining of a modified supervised learning network at each partition. This paper presents work on extracting disjunctive rules by a method which, unlike supervised BRAINNE, does not require resolving several modified networks. Both conjunctive and disjunctive rules are directly obtained from Hybrid BRAINNE.

BRAINNE is designed to deal with both discrete and continuous data. For continuous data, a dynamically chosen threshold value is used to produce bounds for each of the attributes in a conjunctive rule. This paper also examines several improvements to BRAINNE to increase its generalisation capability by:

- using a test set to stop the network from overtraining i.e. memorising;
- defining improved bounds for continuous data.

2 SUPERVISED BRAINNE

2.1 Brief overview of BRAINNE

In this paper, we present a brief overview of the BRAINNE approach for extracting rules and concepts. As a precursor to BRAINNE, the extraction of production rules, higher level concepts and a concept hierarchy from a fully connected, single layer network trained with Hebb's Rule was investigated. Sestito and Dillon (1994) explain that the weights in the weight matrix indicate the contribution each input makes to each output. Thus, the input with the largest weight link to an output makes the largest contribution to that output. This forms the basis for extracting a conjunctive rule for each output using the contributory inputs or attributes in the antecedent. Higher level concepts or specific combinations of defining attributes can be extracted from the initial set of rules. A concept hierarchy can be formed, if possible, with higher lying concepts representing generalisations and lower lying concepts representing specialisations (Sestito & Dillon, 1994). The method requires the repetition of the following steps until all rules are reduced to tautologies (i.e. the concepts imply themselves) (Sestito & Dillon, 1994):

- Select the rules with the least number of contributory attributes.
- Insert these concepts into the rule base by replacing all combinations of the attributes that define a concept by the concept itself, in any of the remaining rules.

This rule extraction approach has some serious limitations. The modelling capacity of single layer neural networks does not extend to complex, nonlinear domains, and the rules produced are conjunctive (Sestito & Dillon, 1994). Domains exist which have two or more different sets of defining attributes for the one concept, requiring a disjunctive rule.

2.2 Supervised BRAINNE and conjunctive rules

To extract conjunctive rules for each output, BRAINNE uses a multi-layered neural net to determine the relevance of inputs and a single layer neural net to determine the irrelevance of inputs to a particular output. A summary of the steps in Sestito and Dillon (1990a-c, 1994) to extract conjunctive rules follows.

- Train a modified, multi-layered neural network with one hidden layer using Back-propagation. The modified net has an extended input set consisting of the original inputs and the desired outputs as additional inputs.
- Determine the contribution of an original input to an output using the Sum of Squares Error (SSE) measurement:

$$SSE_{ab} = \sum_{j} (W_{bj} - W_{aj})^2$$

  where *a* is an input attribute, *b* is an additional input (desired output), and W_{aj} and W_{bj} are the weight links between *a* and hidden unit j, and *b* and hidden unit j, respectively. The smaller the SSE value, the greater the contribution of the original attribute to the output. This is based on the premise that there is a strong association between an input and an additional input (desired output) if their weight links to each hidden unit are similar.
- Negate the original inputs and use these as inputs to a single layer neural network with the desired outputs as its outputs. Train the net using Hebb's Rule to obtain a measure of irrelevance of the original inputs to the outputs. The smaller the weight value, the greater the contribution of the original attribute to the output.
- Calculate the product of the inhibitory weight values and the SSE measurements for all combinations of the inputs *a* and the outputs *b*. The smaller the product value, the greater the contribution of the original attribute to the output.
- For each output, sort the product list from maximum to minimum. Select those input attributes below some clear cut-off point in the product list as the contributory attributes in the antecedent of the corresponding conjunctive rule. A clear cut-off occurs if one of two consecutive products is 2 or 3 times larger than the other.
- From this initial set of rules, extract a concept hierarchy, if required.

2.3 Supervised BRAINNE and disjunctive rules

Two difficulties can be encountered in the process described in Section 2.2. The constructed conjunctive rule may not be unique, or the sorted product list may have no clear cut-off point. In both these cases, the procedure for extracting disjunctive rules is required for that output.

The underlying strategy is to divide the examples in the training set of the particular output into subsets that can be uniquely described by one conjunctive rule. BRAINNE initially splits the examples of the output into subsets using the attribute with the smallest nonzero product value. Note that any attribute with a zero product value is automatically selected as well. Two initial rules are formed. For example, if the sorted products list for OUTPUT1 has attribute a1 with a zero value and attribute a2 with a value of 0.1 as the next largest attribute, then a2 is selected as the splitting attribute and a1 is automatically selected. Two subsets are formed for the output and the two corresponding initial rules are:

a1 AND a2 $\Rightarrow$ (OUTPUT 1)
a1 AND NOT a2 $\Rightarrow$ (OUTPUT 1).

If an initial rule covers no examples it is discarded. Also a check is made to see if an initial rule is stopped from further expansion.

A new modified neural net is constructed with the initial rules as the only outputs. The input set consists of the original inputs and the initial rules as additional inputs. When a rule is used as an additional input in BRAINNE, the input is set to 1 if the example being applied to the net is covered by the rule. Otherwise it is set to 0. Sestito and Dillon (1991b, 1994) describe a cyclic guided *generate-and-test* procedure to extract the disjunctive rule for an output. Note in the above, after each partition a new modified neural network has to be constructed and trained.

The full detailed algorithm using this approach is given in Sestito and Dillon (1991b, 1994).

2.4 Supervised BRAINNE and continuous data

The form of the input in the BRAINNE method so far described is restricted to categorical or linear discrete data. The production rules determined are defined by lists of attributes. This makes the process of testing whether an example is covered by a rule straightforward as (Sestito & Dillon, 1994):

- if *all* the attributes defining a rule are present in an example, then the example is covered by the rule; otherwise it is not covered.

If continuous data is used in systems that require discrete inputs, the solution often adopted is to break the continuous data up into two or more classes or bins by selecting an appropriate number of threshold values. Disadvantages of this

approach include an increase in the number of input attributes and the arbitrary nature of the choice of the number of thresholds and their values (Sestito & Dillon, 1994). One of the positive features of neural networks is their ability to handle continuous data directly without any preprocessing.

BRAINNE uses the normalised values of continuous attributes directly, with one original input for each continuous attribute. This requires an approach to determining if a rule covers an example. A rule is a list of attributes, whereas a continuous attribute in an example is a normalised real number. A method for comparing these different representations is needed when determining the coverage of a continuous example by a rule (Sestito & Dillon, 1992, 1994).

When a rule is used as an additional input in BRAINNE, its value is set to 1 if the example being applied to the net is covered by the rule. Otherwise it is set to 0. To determine coverage for an example with any continuous attributes among its features, the following statistical measure calculating the closeness of the example to the rule is used (Sestito & Dillon, 1994):

$$\text{SumDifference} = \sum_{i} (\text{R:act:attr}_i - \text{E:act:attr}_i)^2$$

where i ranges over all the attributes in the rule (continuous and discrete),

R:act:attr_i is the value of the attribute in the rule, and

E:act:attr_i is the value of the attribute in the example.

If attrC is a continuous attribute that is present in the antecedent of a constructed rule in BRAINNE, it is assigned the value 1.0 in the rule. Similarly, if NOT attrC is in the antecedent, it is assigned the value 0.0. If the SumDifference is less than or equal to a small threshold *t*, the rule covers the example. During the operation of continuously BRAINNE, this threshold is dynamically checked and increased if necessary to ensure that the appropriate number of examples is classified.

The continuous and discrete defining attributes for each rule have been determined by the process described earlier but the possible range of each continuous attribute in each rule is still required. First, all the examples are tested via the SumDifference to obtain the example set covered by the rule. Then the range for each continuous attribute in the rule is obtained directly from the example set. The minimum MinC and maximum MaxC of attrC, say, over the example set determine the bounds of attrC in the corresponding clause in the rule's antecedent:

$$(\text{MinC} \leq \text{attrC} \leq \text{MaxC}).$$

If NOT attrC were in the rule, exactly the same procedure is followed to obtain a clause in identical format to the one above, with an appropriate range.

BRAINNE reclassifies the examples using the final form of the rules with the ranges as determined directly from the data. Several positive features of this approach should be mentioned. The calculation of ranges directly from the data avoids user selection of the ranges. The user may have no idea of appropriate ranges. Also, as the ranges are determined separately for each rule, classification

precision is enhanced, and the question of coverage of an example by a rule is simply answered.

3 IMPROVING THE EXTRACTION OF DISJUNCTIVE RULES USING A HYBRID NEURAL NETWORK

Let us consider the case of an output class O_1 that consists of two disjunctive subclasses, O_{s1} and O_{s2}, each of which is defined by a separate rule, R1 and R2 respectively. These rules have the form:

R1: IF A_1 THEN O_{s1}
R2: IF A_2 THEN O_{s2}.

Class O_1 would then be defined by the disjunction of the antecedents to give:

IF $A_1 \vee A_2$ THEN O_1.

A_1 and A_2 could be composite premises involving the conjunction of several attributes $(a_{11}, \ldots, a_{1n})$ and $(a_{21}, \ldots, a_{2m})$ respectively.

All the attributes $(a_{11}, \ldots, a_{1n}, a_{21}, \ldots, a_{2m})$ that are part of the disjunctive rule would be forced to have a strong link to the output Class O_1 during training by a supervised network. It is difficult to distinguish which of the attributes are a conjunctive group (say a_1) and which groups are disjunctive. This led to the need for the technique described in the previous section involving multiple resolving of a neural network to separate out groups of attributes that are conjunctive with each other and disjunctive with other groups.

An unsupervised network essentially identifies combinations of variables that occur together frequently and allocates them to a class. In the case of the above disjunctive rule, it would group the features $(a_{11}, \ldots, a_{1n})$ to a cluster which we could identify as the subclass O_{s1}, and the features $(a_{21}, \ldots, a_{2m})$ to a cluster which we could identify as the subclass O_{s2}. These clusters corresponding to O_{s1} and O_{s2} would not be identified by the unsupervised approach as belonging to the same output class O_1 as there is no information about the target output classes.

If this information were subsequently used with a supervised network, we could distinguish that the clusters O_{s1} and O_{s2} are disjunctive subclasses of the output class O_1. It was felt, therefore, that a hybrid approach which combines supervised and unsupervised learning would be essential to identify disjunctive rules directly.

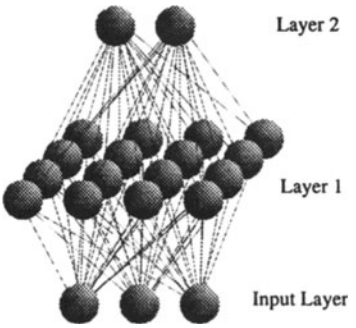


Figure 1 Hybrid neural network.

A hybrid network consisting of two separate layers is required (Figure 1). The Kohonen model (Kohonen, 1990) is used as the unsupervised network in layer 1 due to its simple structure and its ability to extract inherent statistical features. The output from layer 1 is used as the input for layer 2. Layer 2 consists of a supervised network using a single layer perceptron model (McClelland & Rumelhart, 1988). Once layer 1 is trained on the information, it is analysed using the Unsupervised BRAINNE algorithm explained in Section 3.1. This analysis process generates a set of production rules with consequents (C_1, C_2, ..., C_n) corresponding to clusters formed in the Kohonen output layer.

The layer 2 supervised network is then stimulated with the outputs from the Kohonen layer. Once trained, it is used to classify the clusters into different output classes. The example data is really data suitable for supervised training in that, for each example, the values of the inputs as well as the associated target output class are given. Hence the number of output classes is known. Further, we know that a given example corresponds to a given class. Thus, this process replaces the symbolically denoted consequents of the production rules generated by the first unsupervised layer with known output classes. Importantly, this automatically picks up the disjunctive classes.

The algorithm therefore is as follows:

1. Apply Supervised BRAINNE (Sestito & Dillon, 1994). If only unique conjunctive rules are indicated, generate these rules and stop. If not, go to step 2.
2. Train the unsupervised layer 1 on the data and use Unsupervised BRAINNE to generate production rules with symbolically labelled clusters as the consequents of these rules.
3. Run the data through layer 1 and train the supervised network. Replace the virtual labels assigned in step 2 with known output classes.

The method of rule extraction using Unsupervised BRAINNE employed in step 2 is explained in Section 3.1. The method used in step 3 is explained in Section 3.2.

3.1 The unsupervised layer

Unsupervised BRAINNE is a method for extracting production rules that define the clusters in the output layer of a trained Kohonen network. The following is an outline of the procedure (Sestito & Dillon, 1994):

1. Preprocess data to transform it into the correct format.
2. Assign a given dimension to the Kohonen layer.
3. Determine the weights of the Kohonen network.
4. Delete the irrelevant attributes that are identified as having zero weight vector components to all the output nodes.
5. Use either the threshold technique or breakpoint technique to determine the contributory and inhibitory inputs for the initial rules.
6. Using the antecedents of the rules, define clusters of output nodes, each cluster associated with a given rule (see details below).
7. If the number of nodes not belonging to a cluster is large, assign a new dimension to the Kohonen layer and go to step 3.
8. Assign virtual labels to the clusters.

A description of steps 1 to 3 can be found in Kohonen (1990).

In Hybrid BRAINNE the threshold technique is employed in step 5. This technique uses the premise that inputs with the largest weights to an output node contribute most to that node. However, due to problems such as noise inherent in real-world domains, inputs with weights within a certain limit from the maximum weight are considered.

All inputs *i* are included as contributory in the antecedent of the rule for output node *j* in the Kohonen layer if:

$$|W_{maxj} - W_{ij}| < T$$

where the threshold T is a number between 0 and 1. Note that if W_{maxj} is not sufficiently large, no input is considered contributory. Inhibitory inputs are determined by the same means, but they lie within a threshold, T_{inh}, from zero. These inputs are negated in the antecedent. Inputs that are not selected as contributory or inhibitory are ignored.

Clusters are defined as groups of nodes in the Kohonen layer which have the same antecedent. Each cluster is assigned a virtual label: C_1, C_2, ..., C_n. The rules derived from this stage are then of the form

$$x_1 \wedge x_2 \wedge \ldots \wedge x_n \Rightarrow C_i,$$

where x_k is the symbolic equivalent of input k and C_i represents cluster I.

Nodes which do not belong to any clusters could represent:

- intermediate concepts or higher level concepts in a concept hierarchy that is yet to be identified, or
- a mixture of instances of topologically adjacent clusters which does not represent a separate concept.

The first case is identified by the existence of one or more inputs which are neither contributory or inhibitory. Corresponding nodes are designated a new cluster. If the number of nodes which fall into the second case is unacceptably large, a new output layer with different dimensions is specified and the process is repeated until the number of unidentified nodes reduces to an acceptable level.

3.2 The supervised layer

During training in this second layer, we note that the example data is suitable for supervised training in that the target output class for each example is given. Hence, we also know the number of output classes. The first layer constructs the initial conjuncts which form clusters; each node in the Kohonen layer has a conjunctive rule attached to it. A cluster is merely a group of nodes with the same antecedent.

The second layer of the hybrid network is used to construct the disjuncts between the clusters. It consists of a single layer perceptron with every node in the Kohonen layer connected to every node in this layer. Activation of the output node a_j:

$$a_j = f(\sum a_i w_{ij} - T)$$

where a_i is the activation of input node i, and w_{ij} is the weight between nodes i and j, and T is the threshold. The activation function, f, is sigmoid. The delta rule is used for training; the weights are updated according to:

$$\Delta w_{ij} = \varepsilon \delta_j a_i$$

where δ_j, the error for output unit j, is given by $\delta_j = t_j - a_j$, the difference between target t_j and the actual activation. The learning rate, ε, usually lies between 0 and 1. All weights in this layer are initialised to 0 before training.

Training is performed by running the input through the rules produced by Unsupervised BRAINNE, firing the appropriate nodes in the Kohonen layer. These activations are then passed to the output nodes via weights. The error between the actual activation of the output and the desired activation is calculated and the weights in the second layer are adjusted accordingly. This process is repeated until the error is reduced to a predefined tolerance, when the training is stopped.

When a training example is presented to the input of the first layer of the hybrid network, only those clusters in the first layer whose rules match the input are activated (output of 1). All other nodes are deactivated (output of 0). The weights between active clusters and those outputs in the supervised layer whose desired activation is 1 will be incremented, while the weights between the same clusters and the remaining outputs will be decremented. If the clusters are active when a

particular output node is supposed to be active, training will ensure the sum of all the weights from the active clusters to the output node is greater than its threshold:

$$\Sigma a_j w_{ij} > T.$$

This means that, once training is complete, any cluster with positive weights to an output will independently define that output, ensuring that it can form part of a disjunct if other such clusters exist for the same output. Hence, clusters with positive weights to output *j* can be combined in disjunction:

$$C_1 \vee C_2 \vee \ldots \vee C_n \Rightarrow X_j$$

where X_j is the symbolic equivalent to output node j. This is equivalent to substituting X_j, in the conclusion of the rules defining C_1 to C_n. If C_i is in the disjunct for X_j, and

$$x_1 \wedge x_2 \wedge \ldots \wedge x_n \Rightarrow C_i$$

is defined, then the following new rule is added to the production rule set:

$$x_1 \wedge x_2 \wedge \ldots \wedge x_n \Rightarrow X_j.$$

A hierarchy of concepts is then produced by selecting rules with the least number of contributory inputs and replacing matching sections of antecedents in the rule set with the conclusion of the selected rules (Sestito and Dillon 1994). An example is shown in Table 1. The corresponding concept hierarchy is given in Figure 2.

Table 1 Forming a Concept Hierarchy

Original rule set	*After forming hierarchy*
$(a \wedge b) \Rightarrow (X)$	$(a \wedge b) \Rightarrow (X)$
$(a \wedge b \wedge c) \Rightarrow (Y)$	$(X \wedge c) \Rightarrow (Y)$
$(a \wedge b \wedge d) \Rightarrow (Z)$	$(X \wedge d) \Rightarrow (Z)$

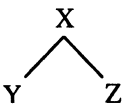


Figure 2 Concept hierarchy.

The rule set may then be reduced by selecting the highest level rules in the concept hierarchy that define only one output.

If there exist outputs that do not have at least one unique rule it is possible that the unsupervised layer is not large enough to capture the information required for defining these outputs. In this case the dimensions of the unsupervised layer may be increased and training repeated.

3.3 Results for extraction of disjunctive rules

LED data

The LED data problem involves seven inputs representing light emitting diodes on a digital display (Figure 3). The training set consists of ten unique examples representing the ten decimal numbers.

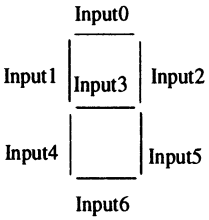


Figure 3 Seven inputs for the LED data problem.

The solution to this problem does not require disjunctive rules, so in order to test the ability of Hybrid BRAINNE to derive disjunctive rules, the training set was modified so that only five output classes exist as shown in Table 2. This means that there could exist six separate rules for output 4.

Table 2 Modified LED Training Set

Actual digit output	*New output*
0	0
1	1
4	2
7	3
2, 3, 5, 6, 8, 9	4

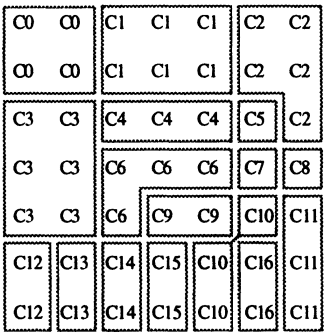


Figure 4 Kohonen layer clusters.

After training the Kohonen layer on the ten training examples the clusters show in Figure 4 were identified. Once the second layer was trained, relevant outputs were added to the conclusion of each rule. The rule set at this stage is listed in Table 3.

A hierarchy of rules was then constructed. The five separate structures are shown in Figure 5. The upper diagram shows where the clusters reside within the hierarchy, while the lower diagram is a representation of the inputs that are on, off or not specified.

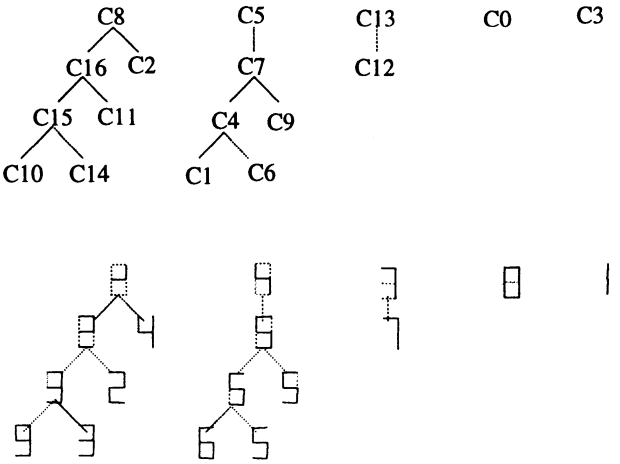


Figure 5 Clusters forming the five separate structures.

Taking the highest level clusters from the hierarchy which do not have clashing outputs results in the rules set given in Table 4. Only two disjunctive rules were required for output 4.

Table 4 Final Rule Set

Trimmed rule set
(Input0∧Input1∧Input2∧¬Input3∧Input4∧Input5∧Input6) ⇒ (Output0)
(¬Input0∧¬Input1∧Input2∧¬Input3∧¬Input4∧Input5∧¬Input6) ⇒ (Output1)
(¬Input0∧Input1∧Input2∧Input3∧¬Input4∧Input5∧¬Input6) ⇒ (Output2)
(Input0∧¬Input1∧Input2∧¬Input3∧¬Input4∧Input5∧¬Input6) ⇒ (Output3)
(Input0∧Input1∧Input3∧Input5∧Input6) ⇒ (Output4)
(Input0∧Input2∧Input3∧Input6) ⇒ (Output4)

4 IMPROVING THE GENERALISATION CAPABILITY

4.1 Improvement of generalisation capability by avoiding overtraining

When a network is overtrained its generalisation capabilities decrease. A properly trained network can respond confidently with data it has not seen previously. An overtrained network, however, tends not to classify unseen data properly, while it fits the training data very well. An overtrained network is said to have memorised the training pattern. Thus an overtrained network guarantees that when the network is provided with a pattern from the training set, it will respond exactly in the same manner that it was trained (El-Sharkawi, 1996).

A network is said to have good generalisation capability when it can respond well with data it has not seen before. There are several different ways to ensure that a network is trained, and has not just memorised. All these techniques are based on the fact that, for given training and test data, a network should respond with a similar error for the training and test data sets (El-Sharkawi, 1996).

For BRAINNE, the training set is divided into two subsets (one larger than the other). The larger subset is used to train the network and the smaller subset is used to monitor the error, and this is used as the basis of the stopping criterion.

After each iteration of the network, the test data is used to calculate the error measures of the network. We then try to determine the global minimum for the test data error and stop training the network at that point. The training error usually decreases as the number of epochs of training increases. However, the test error decrease goes through a minimum and then increases.

Table 5 Example Test Error Values

Table 3 Initial Rule Set

Initial rules
(Input0∧Input1∧Input2∧¬Input3∧Input4∧Input5∧Input6) ⇒ (C0∧Output0)
(Input0∧Input1∧¬Input2∧Input3∧Input4∧Input5∧Input6) ⇒ (C1∧Output4)
(¬Input0∧Input1∧Input2∧Input3∧¬Input4∧Input5∧¬Input6) ⇒ (C2∧Output2)
(¬Input0∧¬Input1∧Input2∧¬Input3∧¬Input4∧Input5∧¬Input6) ⇒ (C3∧Output1)
(Input0∧Input1∧¬Input2∧Input3∧Input5∧Input6) ⇒ (C4∧Output4)
(Input1∧Input3∧Input5) ⇒ (C5∧Output2∧Output4)
(Input0∧Input1∧¬Input2∧Input3∧¬Input4∧Input5∧Input6) ⇒ (C6∧Output4)
(Input0∧Input1∧Input3∧Input5∧Input6) ⇒ (C7∧Output4)
(Input2∧Input3) ⇒ (C8∧Output2∧Output4)
(Input0∧Input1∧Input3∧¬Input4∧Input5∧Input6) ⇒ (C9∧Output4)
(Input0∧Input1∧Input2∧Input3∧¬Input4∧Input5∧Input6) ⇒ (C10∧Output4)
(Input0∧¬Input1∧Input2∧Input3∧Input4∧¬Input5∧Input6) ⇒ (C11∧Output4)
(Input0∧¬Input1∧Input2∧¬Input3∧¬Input4∧Input5∧¬Input6) ⇒ (C12∧Output3)
(Input0∧¬Input1∧Input2∧¬Input4∧Input5) ⇒ (C13∧Output3)
(Input0∧¬Input1∧Input2∧Input3∧¬Input4∧Input5∧Input6) ⇒ (C14∧Output4)
(Input0∧Input2∧Input3∧¬Input4∧Input5∧Input6) ⇒ (C15∧Output4)
(Input0∧Input2∧Input3∧Input6) ⇒ (C16∧Output4)

Epoch	*Test Error*
11	4.32
12	4.20
13	4.50
14	4.30
15	4.10
16	4.80
17	5.30
18	5.58

For BRAINNE, we compared the test errors from one iteration to the next. If the error decreases and then starts to increase consistently the training is stopped at the point where the error starts to increase. For example, for the error values illustrated in Table 5 and Figure 6, the training is stopped after epoch 15 since the error value increases consistently after that point.

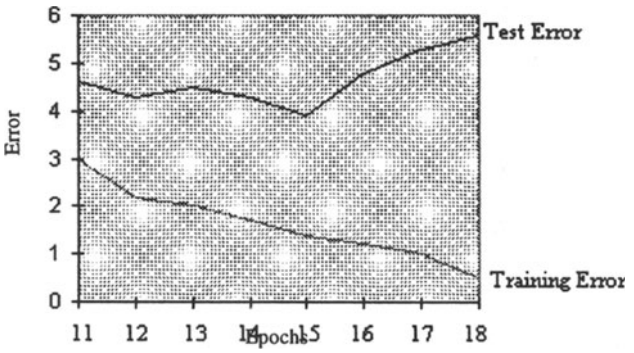


Figure 6 Example test and training errors.

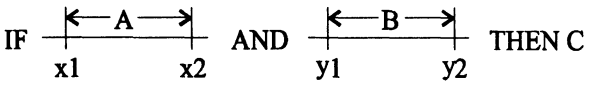
By avoiding overtraining, the generalisation capability of BRAINNE should increase, thus allowing better performance with previously unseen data.

4.2 Improvement of bounds

For continuous data, BRAINNE produces conjunctive rules where the attributes usually have an upper bound and a lower bound. For instance, the rule:

IF $(x1 \leq a \leq x2)$ AND $(y1 \leq b \leq y2)$ THEN C

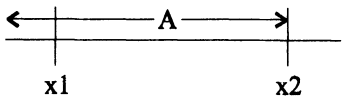
can be represented as:



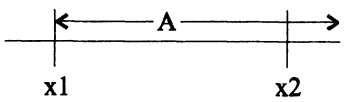
where $\{x1, x2 \in X : X \in \Re\}$ and $\{y1, y2 \in Y : Y \in \Re\}$.

These bounds are created during training using the training data. They reflect the upper and lower values within the training set for a contributory input. The problem with this approach is that the bounds may be too rigid for a new data set. We could have a situation where the unseen data range may fall outside the bound generated by BRAINNE. The following can happen:

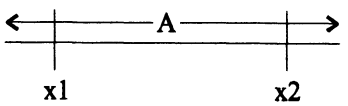
- Lower range of the new data set may fall outside the generated bound.



- Upper range of the new data set may fall outside the generated bound.



- Both upper and lower ranges may be outside the generated bound.



The most obvious way to overcome this problem would be to increase the bounds towards the maximum ([0,1] for normalised data) for a contributory input. This however, can present us with new problems of misclassification. BRAINNE is tested for two attributes:

1. coverage of data (i.e. % of examples covered by the rules generated by BRAINNE);
2. correctness of classification.

To increase the generalisation capability we need to increase the coverage of unseen data. However, we also have to ensure that we do not increase the misclassification rate in the process. It is possible to have 100% coverage of the unseen data, but it is very likely that the network will have a much higher occurrence of misclassification.

Using the approach given in Sestito and Dillon (1994) an initial set of bounds for the contributory inputs is generated. We then increase these bounds progressively, checking for misclassification after each increment. If an increase leads to a misclassification, the bound is returned back to its previous value. The minimum bound and the maximum bound for each attribute were increased separately.

The steps for improvement in bounds are as follows:

1. Increase maximum/minimum bound for an attribute by θ (usually around 10%).
2. Check for misclassification.
3. If classification error is produced, decrease bound by θ else repeat step 1.
4. Pick next bound and go to step 1. If all bounds for all attributes have already been processed, stop.

This method is used for each contributory attribute of a rule. For normalised data, the largest value for the upper bound is 1 and the smallest value for the lower bound is 0.

4.3 Results for the generalisation capability

The methods discussed in Sections 4.1 and 4.2 were tested using the IRIS domain from the Irvine machine learning databases.

The rules generated by the system were tested for coverage and classification capability using the training data, overtraining test data and unseen test data. The data contains 3 different types of plants described by 4 input attributes:

Table 6 Iris Domain

Input Attributes	*Outputs*
sepal-length	Iris Setosa
sepal-width	Iris Virginica
petal-length	Iris Versicolor
petal-width	

The input attributes are normalised using the formula:

$$\text{Normalised Value} = \frac{\text{Actual Value - Min}}{\text{Max - Min}}.$$

This domain consists of 150 examples. Of these examples, 90 were used to train the network, 30 to test for overtraining and the remaining 30 (unseen) examples were used to test the performance of the rules generated by the system. The network was trained with 6 hidden units, learning rate of 0.2 and momentum value of 0.7. The results are summarised in Tables 7 to 9.

Table 7 BRAINNE

Data Set	*Classification Incorrect*	*Covered/Not Covered*
Training Data	3	76/14
Test Data	0	26/4
Unseen Data	1	21/9

Table 8 BRAINNE after Overtraining Check

Data Set	*Classification Incorrect*	*Covered/Not Covered*
Training Data	3	77/13
Test Data	0	26/4
Unseen Data	1	23/7

Table 9 BRAINNE after Increase in Bounds

Data Set	*Classification Incorrect*	*Covered/Not Covered*
Training Data	3	87/3
Test Data	0	28/2
Unseen Data	1	29/1

A slight improvement in the coverage capability after using the overtraining test can be seen in Table 8. From Table 9, there is a significant improvement in the generalisation capability of BRAINNE without any reduction in classification capabilities.

5 CONCLUSION

This paper presented three different sets of improvements to the BRAINNE method. The first of these uses a two layer hybrid network, with unsupervised learning at the first layer and supervised learning at the second, to extract disjunctive rules directly. This avoids the need in the previous approach for multiple resolving of modified networks corresponding to different sets produced by a generate-and-test procedure.

The second approach stops training earlier by monitoring the error measures using a separate test set of examples. This improves the generalisation capability of the neural network and of the derived rules.

The third improvement leads to progressive extension of the bounds used in defining rules for the case of continuous features. All of these approaches were tested on different sets of examples and improvements were obtained.

6 REFERENCES

Bloomer, W.F., Dillon, T.S. and Witten, M. (1996) Hybrid BRAINNE: A method for developing symbolic rules from a hybrid neural network. *IEEE International Conference on Systems Man and Cybernetics*. Beijing China. October. pp. 14-17.

Craven, M.W. and Shavlik, J.W. (1993) Learning symbolic rules using artificial neural networks. *Proceedings of the Tenth International Conference on Machine Learning*. Amherst, MA. pp. 73-80.

Dillon, T.S., Sestito, S., Witten, M. and Suing, M. (1993) Automated knowledge acquisition using unsupervised learning. *Proceedings of the Second IEEE Workshop on Emerging Technology and Factory Automation (EFTA '93)*. Cairns. September. pp. 119-28.

Dillon, T.S., Sestito, S., Witten, M. and Suing, M. (1994) Symbolic knowledge from unsupervised learning. *International Symposium on Integrated Knowledge and Neural Heuristics*. Florida USA. May. pp. 47-56.

El-Sharkawi, M.A. (1996) Neural networks and its ancillary techniques as applied to power systems. In *IEEE Tutorial Course on Artificial Neural Networks with Application to Power Systems*. (eds M.A. El-Sharkawi and D. Niebur).

Fu, L. (1991) Rule learning by searching on adapted nets. *Proceedings of the Ninth National Conference on Artificial Intelligence*. Anaheim, CA. pp. 590-5.

Gallant, S.I. (1993) *Neural Network Learning and Expert Systems*. MIT Press, Cambridge, MA.

Kohonen, T. (1990) The self organising map. *Proceedings of the IEEE*, 78(9) September, 1464-80.

McClelland, J.L. and Rumelhart, D.E. (1988) *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs and Exercises.* MIT Press, Cambridge Massachusetts. pp. 83-137.

Saito, K. and Nakano, R. (1988) Medical diagnostic expert system based on PDP model. *Proceedings of the IEEE International Conference on Neural Networks.* San Diego, CA. pp. 255-62.

Sestito, S. and Dillon, T.S. (1989) Using neural networks for the extraction of high level knowledge representation for machine learning. *Australian Artificial Intelligence Conference (AI '89).* Melbourne Australia. pp. 413-28.

Sestito, S. and Dillon, T.S. (1990a) Using sub-symbolic methods for machine learning of high level knowledge representation. Keynote paper. *Finnish AI Symposium (SteP-'90).* Oulu Finland. pp. 27-49.

Sestito, S. and Dillon, T.S. (1990b) Using multi-layered neural networks for learning symbolic knowledge. *Australian Artificial Intelligence Conference (AI '90).* Perth Australia. November. pp. 249-62.

Sestito, S. and Dillon, T.S. (1990c) Machine learning using single layered and multi-layered neural. *IEEE Conference on Tools for AI (TAI-'90).* Washington D.C. November. pp. 269-75.

Sestito, S. and Dillon, T.S. (1991a) Using single-layered neural networks for the extraction of conjunctive rules and hierarchical classifications. *Journal of Artificial Intelligence.* 1 November, 157-73.

Sestito, S. and Dillon, T.S. (1991b) The use of sub-symbolic methods for the automation of knowledge acquisition for expert systems. *Eleventh International Conference on Expert Systems and their Applications (Avignon '91).* Avignon France. pp. 317-28.

Sestito, S. and Dillon, T.S. (1992) Automated knowledge acquisition of rules with continuously valued attributes. *Twelfth International Conference on Expert Systems and their Applications (Avignon '92).* Avignon France. pp. 645-56.

Sestito, S. and Dillon, T.S. (1993) Knowledge acquisition with neural networks. In *Modern Tools for Manufacturing Systems.* Elsevier Science Publishers BV, Nederland. pp. 149-76.

Sestito, S. and Dillon, T.S. (1994) *Automated Knowledge Acquisition.* Prentice Hall, Sydney.

Towell, G. and Shavlik, J. (1993) Extracting refined rules from knowledge-based neural networks. *Machine Learning.* 13(1), 71-101.

7 BIOGRAPHY

Tharam S. Dillon received a Bachelor of Engineering (Honours) in 1967 and PhD in 1974 from Monash University, Melbourne, Australia. He is currently Professor of Computer Science and Computer Engineering at La Trobe University, and Director of the Applied Computing Research Institute. He is a Fellow of the IEEE (USA), the Institution of Engineers (Australia), and the Safety and Reliability Society (UK). He has published over 320 papers in international and national journals and conferences and written three books and edited four other books. He is

editor-in-chief of the *International Journal of Computer Systems Science and Engineering* and the *International Journal of Engineering Intelligent Systems*, as well as co-editor of the *Journal of Electric Power and Energy Systems* and an associate editor of *IEEE Transactions on Neural Networks.*

Tanveer Hossain received a Bachelor of Computer Science (Honours) in 1993 from La Trobe University, Melbourne, Australia. She is currently working on her PhD in the area of automated knowledge acquisition and datamining.

Warren Bloomer received a Bachelor of Computer Science (Honours) in 1995 majoring in computer science and commerce from La Trobe University, Melbourne, Australia. He is currently working on his PhD in the area of automated knowledge acquisition and datamining.

Mary Witten received a Bachelor of Science in 1973 and a Diploma in Education in 1974 from the University of New England, Armidale, Australia. She received a Diploma in Computer Science in 1989 and a Graduate Diploma in Advanced Computer Science in 1996 from La Trobe University, Melbourne, Australia. She is currently working as a research assistant in the Expert and Intelligent Systems Laboratory of the Applied Computing Research Institute at La Trobe University.